8. Distributed training of neural networks with Apache Spark

Author(s)

Name	Institution	Mail Address
Matteo Migliorini	INFN Sezione di Padova	matteo.migliorini@pd.infn.it

General Information

ML/DL Technologies	Feedforward neural network
Science Fields	High Energy Physics
Difficulty	Low
Language	English
Type	Runnable

Software and Tools

Programming Language	Python
ML Toolset	BigDL
Additional libraries	BigDL, Spark
Suggested Environments	INFN-Cloud VM, Spark Cluster

Short Description of the Use Case

The effective utilization at scale of complex machine learning techniques for HEP use cases poses several technological challenges, most importantly on the actual implementation of dedicated end-to-end data pipelines. In this paper we presented a possible solution to this challenges built using industry standard big data tools, such as Apache Spark. In the presented pipeline we exploited Spark in all the steps, from ingesting and processing ROOT files containing the dataset to the distributed training of the model.

In this use case we focus on the training of one of the classifiers using Spark and Intel BigDL, an open source deep learning library from Intel written on top of Spark hence allowing easy scale out computing. Furthermore, BigDL makes it easy to distribute model training between the workers, allowing to obtain an almost linear speedup in the training time. All this complexity is hidden from the user, which is exposed only to an API similar to Keras/Pytorch.

In this simple example, we will train a deep neural network with the goal of classifying three different kind of events. A better description of the HEP use case and model used is provided in the original paper.

How to execute it

A Spark cluster is required in order to perform the training. This can be created on INFN Cloud by performing the following steps:

1. From the INFN Cloud dashboard, click on configure under "Spark + Jupyter cluster"
2. From there, create a cluster with the number of workers needed, e.g. 2 slaves. For this example I've used one master and two slaves with flavor "Large"
3. After the deployment is completed, the cluster can be found under the voice "Deployment". From there retrieve the Jupyter hub endpoint and by clicking on it create a Jupyter server with flavour "Medium".
4. Once the server has been created, it is possible to upload the notebook with the training example which can be found either here or in the attachments.

A subset of the dataset has already been uploaded on minio and should be accessible by everybody. The full dataset used in this example can be downloaded from here and uploaded on a workspace on minio.

In the notebooks, the first three paragraphs, are needed to respectively

1. Get the credentials to access minio
2. Download BigDL packages
3. Create a Spark session

In principle they should not be modified and work out of the box. If one wish to use more workers for the training and slaves ara available, the number of executors can be increased in the spark session in the option *"spark.executor.instances"* and *"spark.cores.max"* (which should be equal to executor.cores*executor.instances).

References

- A detailed description of the work around this example can be found in this paper:
  Migliorini, M., Castellotti, R., Canali, L. et al. Machine Learning Pipelines with Modern Big Data Tools for High Energy Physics. Comput Softw Big Sci 4, 8 (2020). https://doi.org/10.1007/s41781-020-00040-0
- Data and code used for this work can be found in this repository.
- Related blog entries:
  - Machine Learning Pipelines for High Energy Physics Using Apache Spark with BigDL and Analytics Zoo
  - Distributed Deep Learning for Physics with TensorFlow and Kubernetes
- Presentations:
  - Deep Learning Pipelines for High Energy Physics using Apache Spark with Distributed Keras on Analytics Zoo

Attachments